Machine Learning in the calibration process of Discrete Particle Model: The case with Angle of Repose

Quang Hung Nguyen

Head Supervisor: Thomas Weinhart, Daily Supervisor: Anthony Thornton, Additional member: Chen Kuan.

Multi-Scale Mechanics Group, Faculty of Engineering Technology, University of Twente.

1 Introduction

Figure 1: Examples of Granular Materials.[1]

Granular material is a family of material characterized by its large bulk of densely packed particles, ranging from nanometers to centimeters [2], and is able to resist deformation and form heaps, i.e., behave like a solid and withstand strong shear force [3]. Simple examples of granular materials include sand, gravel, clays, seeds, nuts, and all ranges of powders such as coffee powder, cement powder, which is shown in figure 1. Furthermore, many processes and equipments in chemical plants use granular materials, such as catalysis, adsorption, and heat exchangers. Granular materials are projected to make about half of the products and three-quarters of the raw materials used in the chemical industry [4]. Thus, understanding how granular materials behave is of great significance.

The simulation of granular material's bulk mechanical behavior is done using Discrete Particle Model (DPM, or Discrete Element Method - DEM), which generates the movement of individual particles to capture the macro-scale behavior. The DPM is a family of numerical methods for computing the motion of a large number of particles [5], first proposed by Cundall and Strack in the 1970s [6]. Since the properties of granular materials differ wildly, these simulations require an extensive calibration process designed individually for each type of granular material. Some parameters of the granular material model can be measured directly, such as size distribution or density. However, other parameters are effective parameters (i.e., they result from a simplified particle model) and thus cannot

be directly measured. These parameters are then calibrated by choosing a few standard calibration setups (rotating drum, heap test, ring shear cell) and simulating these setups in a DPM simulation, and the missing parameters are determined such that the response of the experimental and simulation setups match.

Recently, coupled with the raise of Machine Learning in other fileds, it has also been applied to solve the calibration problem. This has been done using a Neural Network [7, 8, 9, 10], Genetic Algorithm [11], and a recursive Bayesian sequential Monte-Carlo filtering algorithm named GrainLearning [12]. In this Assignment, two Machine Learning algorithms will be discussed: Neural Network and GrainLearning.

These two algorithms set to treat the calibration problem in two different ways, and likewise, solve it in two different ways: While GrainLearning looks to identify the microparameters from the experimental and DEM simulations's bulk parameters (inverse problem), the Neural Network will help generating a database that can map different microparameters combinations to their corresponding bulk parameters, generated by DEM simulations. In other word, NN will learn the built-in relationship between the micro- and macroparameters of the Discrete Particle Model, thus allow a much faster prediction compare to a full DEM simulation. One advantage of GrainLearning compares to other Machine Learning algorithms such as Neural Network is that it is an unsupervised learning algorithm, i.e., it can starts calibrating with a minimal amount of input information. A normal calibration routine, currently implemented in MercuryDPM would only need the measurements data, parameter range, and the importance weight of each measurement (depends on the modeller's knowledge). Meanwhile, the current approach mentioned in [7, 8, 9] would requires modeller to define a different NN model for each material, train it using a set of DEM simulations, and then validate the correct combinations of input-output by experiment. And although this process can be automated, to date the author has not been aware of any study implemented a fully-automated calibration routine using Neural Network.

todo here: A paragraph describes each section

2 Simulation Method

2.1 Angle of Repose

Angle of Repose (AoR) is one of the most important bulk parameters to describe the characteristics of Granular Material. Despite having a generic name, definitions of AoR differs vividly, subject to different applications and behaviors [13]. In the context of this Assignment, one type of Angle of Repose will be discussed: Static Angle of Repose.

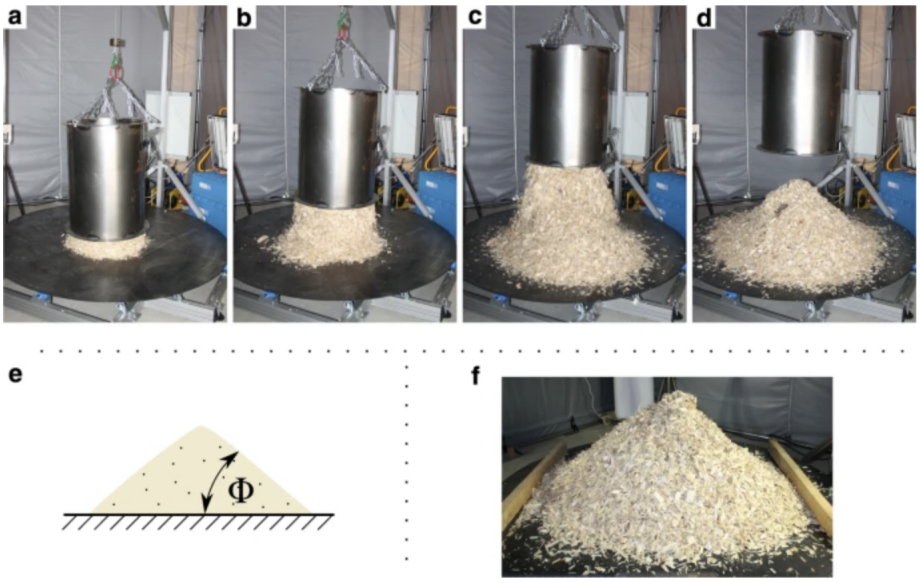

Figure 2: Static Angle of Repose measurement steps [14].

Static AoR, as described in Fig. 2, is defined as the angle that granular solids forms when it piled with a flat surface, and is essential to characterise the coarseness and smoothness of materials. This in turn can help designing a process involved with the material - lower static AoR implies more flowable and thus easier to transport with less energy [15].

2.2 Discrete Particle Model

Discrete Particle Model simulates particle motion by applying forces and torques which derives from particle-particle interactions and external influences, on the basis of the given contact law. It performs calculations of kinematics that a given particle i exerts on other particle j, for each particle in the system, among with the peripheral factors such as gravity and walls. To achieve this results, the particles are assumes to be (1) undeformable - deform therefore implemented as overlap, (2) unbreakable, (3) all internal interactions are due to particle-particle interaction, (4) Each particle pair i, j has only on/viewer.htmle contact point c_{ij} which the forces and torques act on, and (5) all external forces and torques are either body forces and torques or by interacting with a wall [16].

2.2.1 Contact Laws

For each particle i on the system, Eq. 1 describes the internal and external forces, and Eq. 2 describes the torque acting on it [16]:

$$F_i = \sum_{j=1}^{n_p} F_{ij} + \sum_{k=1}^{n_w} F_{ik}^w + F_i^b \tag{1}$$

$$\tau_i = \sum_{j=1}^{n_p} r_{ij} F_{ij} + \tau_{ij} + \sum_{k=1}^{n_w} r_{ik} F_{ik}^w + \tau_{ik}^w + \tau_i^b \tag{2}$$

With F_{ij} interparticle forces, F_{ik}^w the interaction force between each wall and the particle, n_p = number of particles, n_w = number of walls, F_i^b body forces i.e., gravity, and r_{ij} as the branch vector, which connects the particle position r_i with the contact point c_{ij} . The same hold for torques equation, with τ as torque.

The contact law used in the simulations is the Linear Spring-Dashpot model, which is implemented in MercuryDPM as `LinearViscoelasticSpecies`. It defines the interaction between two particles i and j as a damped harmonic oscillators [17]:

$$F_{ij}^n = \begin{cases} k_n \delta_{ij}^n + \gamma_n v_{ij}^n & \text{if } \delta_{ij}^n > 0, \\ 0 & \text{else,} \end{cases} \tag{3}$$

In this equation, $k_n > 0$ represents spring stiffness, $\gamma_n \geq 0$ represents the damping coefficient, v_n the normal vector, and δ_{ij}^n is the overlap between the particles. Two particles interact with each other if and only if they overlap. This contact model is simple, has an analytic solution and less computationally expensive [18], while also suitable for large particles [16].

2.2.2 Angle of Repose measurement

In MercuryDPM, Static AoR is measured by a hollow cylinder simulation, which consists of two cylinders (instead of one cylinder and one plate in Fig. 2). The starting conditions of the simulation are described in Table **X**. For simplicity, all particles in the simulation are assumed to be a perfect sphere. After all the particles are poured into the cylinder, it is let to rest until the bulk's kinetic energy is less than 1% compared to the potential energy (steady-state condition). At this point, the top cylinder is removed, and all the particles that have fallen out of the bottom cylinder below $z = 0$ are deleted. This will result in a cone-shaped heap of particles and a drastic increase in kinetic energy due to gravity. The heap will be rested again until it reach a steady-state condition as mentioned above, and then Static AoR can be measured. The measurement will be done twice, since the kinetic

energy of the system can be increased again after the first measurement. Figure 3 demonstrates an example simulation on MercuryDPM.

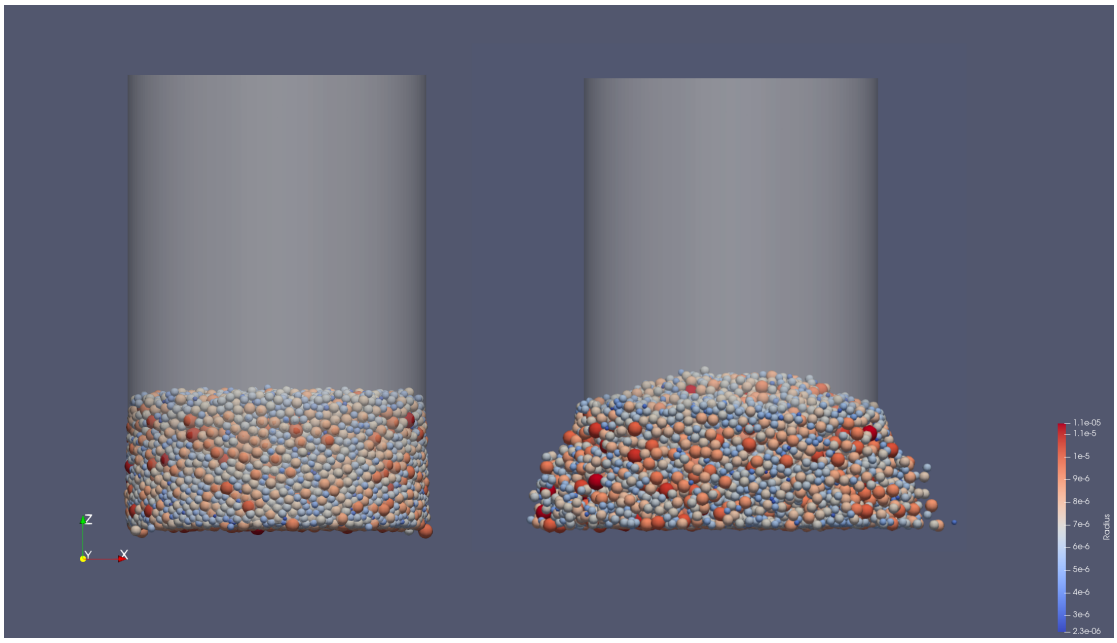

Figure 3: Angle of Repose simulation on MercuryDPM.

2.3 Material and simulation properties

Diameter (µm)	Cumulative Distribution (%)
300	6.21
425	24.50
500	50.55
600	100

Table 1: Particle Size Distribution of Eskal 150

Diameter (µm)	Cumulative Distribution (%)
97	10
138	50
194	90

Table 2: Particle Size Distribution of quartz sand

The above tables will be fixed.

The experimental data described above will be used as constant input values for each DEM simulation. In addition, four variables will be tested to determine their respective static AoR:

- Restitution coefficient: Ratio between the velocity of the particle before and after collision.
- Sliding friction: Force that acts on the opposite direction of the movement between two particles when they collide.
- Rolling friction: Force resists the rolling motion of the particle.
- Bond number: Ratio between gravitational force that act on the particle and the surface tension force, i.e., the easiness of movement due to gravity.

3 Calibration of Discrete Particle Model

3.1 GrainLearning

GrainLearning is a calibration toolbox developed by Cheng et al., utilizes the recursive Bayesian algorithm to estimate the uncertainty parameters in DPM. Initially, a wide range of parameter space is quasi-randomly sampled from the initial guess range to create a prior distribution of each parameter. Then, conditioned on the experimental values, the posterior distribution of the parameters is updated recursively by Sequential Monte-Carlo Filtering (SMC Filter) and fitted to a Gaussian Mixture Model. This process is done iteratively, until a desired value that minimises the loss function is reach, typically 3 iterations. Algorithm 1 and the following sections will describe in brief the calibration workflow implemented in GrainLearning.

Algorithm 1 GrainLearning

Input:
y: Experimental values
$\mathbf{x} = \mathbf{F}(\Theta)$: DEM solver
$(\Theta_{min}, \Theta_{max})$: Initial guess range
Main:
▷ *Set uniform prior distribution:* $p(\Theta) = \mathcal{U}(\Theta_{min}, \Theta_{max})$
for k in range $(0, K)$ **do**
▷ *Sampling parameters:*
if $k = 0$ **then** sample N_p parameters values from initial distribution: $\Theta_k^{(i)} \sim p_0(\Theta)$
else if $k > 0$ **then** sample N_p parameters values from prior distribution: $\Theta_k^{(i)} \sim p_{k-1}(\Theta \mid y_{1:T})$
▷ *Evaluate DPM:* $x_k^{(i)} = F(\Theta_k^{(i)})$
▷ Optimizing σ:
while True **do**
▷ *Compute likelihood (Eq. 4):* $p(y_t \mid \Theta_k^{(i)}) \propto \mathcal{N}(y_{kt} \mid x_{kt}^{(i)}, \Sigma)$
▷ *Compute posterior distribution of of* $\Theta_k^{(i)}$ *conditioned to y (Eq. 5).*
▷ *Compute Effective Sample Size (ESS) with Eq. 6.*
▷ *Stop if target ESS value is reached:*
if $k = 0$ and $ESS > 20\%$ **then** break
else if $k > 0$ and $ESS \sim ESS_{max}$ **then** break
▷ *Fit sampled posterior distribution to Gaussian Mixture Model:* $p(\Theta \mid y) = \sum_\alpha^k \lambda_\alpha \mathcal{N}(\mu_\alpha, \sigma_\alpha)$
▷ *Set new prior distribution:* $p(\Theta) \leftarrow p(\Theta \mid y)$
Output: Θ_{opt} in Θ_K that minimizes $|F(\Theta_{opt} - y)|$

3.1.1 Posterior distribution calculation

Initially, the measurement is assumed to have an error represented by a covariance matrix $\Sigma_\alpha = \sigma\omega_\alpha y_\alpha$, with σ the covariance parameter, and important weight of the measurement ω. With Σ, the likelihood of a given state $\Theta_k^{(i)}$, i.e., the probabilistic prediction to the experimental data y can be estimated by the multivariate normal distribution, with y_t measurement data at time step t, and d the dimension of the state vector $\Theta_k^{(i)}$:

$$p(y_t \mid \Theta_k^{(i)}) \propto \frac{1}{(2\pi)^{d/2}|\Sigma|} exp\left(-\frac{1}{2}(y_{kt} - x_{kt}^{(i)})^\intercal \Sigma^{-1}(y_{kt} - x_{kt}^{(i)})\right) \tag{4}$$

With the calibration system being modelled as a hidden Markov model, the posterior distribution of $\Theta_k^{(i)}$ can be calculated using recursive Bayes' rule:

$$p(\Theta_k^{(i)} \mid y) \propto \prod_{t=1}^{N_t} p(y_t \mid \Theta_k^{(i)}) p(\Theta_k^{(i)}) \tag{5}$$

3.1.2 Effective multi-level sampling

The Effective Sample Size (ESS) is calculated by summing the posterior distribution squared of all the sampled parameters value N_p:

$$ESS = \frac{1}{N_p \sum_{i=1}^{N_p} p(\Theta_k^{(i)} \mid y)^2} \tag{6}$$

The main idea is to draw the sample from the previously acquired knowledge about the relationship between Θ and y. In the first iteration ($k = 0$), The uniform prior distribution is chosen as the proposal density, and the parameter spaces are drawn from there. Subsequently, for $k > 0$, the proposal density will be the posterior distribution from the previous iteration $p(\Theta \mid y)$. After each iterations, the sampling space will get narrower - therefore, to ensure a proper proposal density for the sampling of parameters, the optimization process will be continued until appropriate σ which maximizes ESS is reached.

3.1.3 Identification of microparameters with GrainLearning

In the first iteration of calibration, GrainLearning will initialize a set of parameter combinations using Halton sequence, from the initial guess range provided. This set of parameters will be passed to MercuryDPM to analyze with a Heap test, after which a static AoR is produced. From this data, GrainLearning will compute the next set of parameters on the basis of the previous MercuryDPM output, according to algorithm 1. For each attempt, GL will be running for 4 iterations - except when the simulations of that iteration takes more than two days, and that attempt will be classified as failed.

3.2 Neural Network

Artificial Neural Network (ANN) is a set of algorithms that seeks to identify correlations in data utilizing a technique inspired by how the human brain operates - mimicking how each neuron in the brain signals each other. The most basic ANN model is the Feed-forward Multilayer Perceptron Neural Network (MLPNN), in which the purpose is to define the mapping between the input and output $y = f(x; \theta)$ and approximate the parameter θ which results in the best possible function. In MLPNN, the data will flows in one direction from the input to the output, hence the name feed-forward. Like other supervised learning algorithms, an MLPNN needs to be trained before accurately describing the input and output relations. This is typically done by feeding the network with pre-labeled data, comparing the model's output with the desired output, and updating the weights parameter θ - a process called backpropagation. In this assignment's context, Neural Network (NN) will be used when referring to Feedforward Multilayer Perceptron Neural Network, and NN models implemented in this research are provided by the open-source library `TensorFlow` [19].

3.2.1 Designing a Neural Network

There are no general rules for determining the number of layers and the number of neurons per layer, and it depends heavily on each use case. While Benvenuti et al. [7], He et al. [8], and Daniel et al. [9] used only a single-layer ANN and varied the number of neurons, Ye et al. [10] vary both. However, the ultimate goal in both case is to find the combinations which result in the minimum error while also avoiding overfitting, i.e., the model excels on training but perform poorly on the validation step. In this case, for each simulation material, 250 models ranging from 2 to 15 layers and 5 to 15 neurons per layer are tested to determine the best model. Each model is trained for 50 epochs with a batch size of 32, and the metric used to grade the model is Mean Absolute Error. In addition, 20% of the data will be saved for validation of the model.

Another important component of a Neural Network is the activation function. Since each neuron performs calculation by multiplying the input with weight and adding a bias, the activation function's role would be introducing a non-linearity element into an otherwise linear neuron. According to Goodfellow et al., [20], Rectified Linear Unit (ReLU) is the recommendation for most Deep Learning models, with its ability to preserve much of the properties due to its near-linear shape. ReLU activation function is defined as $f(x) = max(0, x)$.

3.2.2 Training and evaluation method

To train the Neural Network model, 500 DEM simulations with randomized combinations of input parameters have been performed, in order to create a database which maps DEM microparameters to static AoR. Other 125 simulations **did not finish on the time constraint set, and as a results marked as an inaccurate combination.** Afterwards, over 1,000,000 different combinations, as describe in table 3, will be processed by the NN model. Any combinations that producess a static AoR within the 0.1% margin of error to the experimental data is marked as a correct combination. This combination will then be evaluated independently by a DEM simulation, to verify the ability of NN model correctly describes bulk behavior of a material based on the given contact law.

	Restitution Coefficient	Sliding Friction	Rolling Friction	Bond number
Range	[0.5 1]	[1e-5 1]	[1e-5 1]	[1e-5 1]
Number of values	30	30	30	30

Table 3: Random evenly-spaced microparameters combinations

3.3 Random Forest Regressor

4 GrainLearning evaluation

Material	Quartz Sand			Limestone		
Attempt	1	2	3	1	2	3
Restitution Coefficient	[0 1]	[0 1]	[0 1]	[0 1]	[0 1]	[0 1]
Rolling Friction	[0 1]	[0 0.5]	[0.5 1]	[0 1]	[0 0.5]	[0.5 1]
Sliding Friction	[0 1]	[0 0.5]	[0.5 1]	[0 1]	[0 0.5]	[0.5 1]
Bond number	[0 1]	[0 0.5]	[0.5 1]	[0 1]	[0 0.5]	[0.5 1]

Table 4: Calibration attempts using GL

4.1 Limestone

In the first and second calibration set of limestone, the clustering ability of GrainLearning is demonstrated - especially on the second one, as shown on Fig. 4.

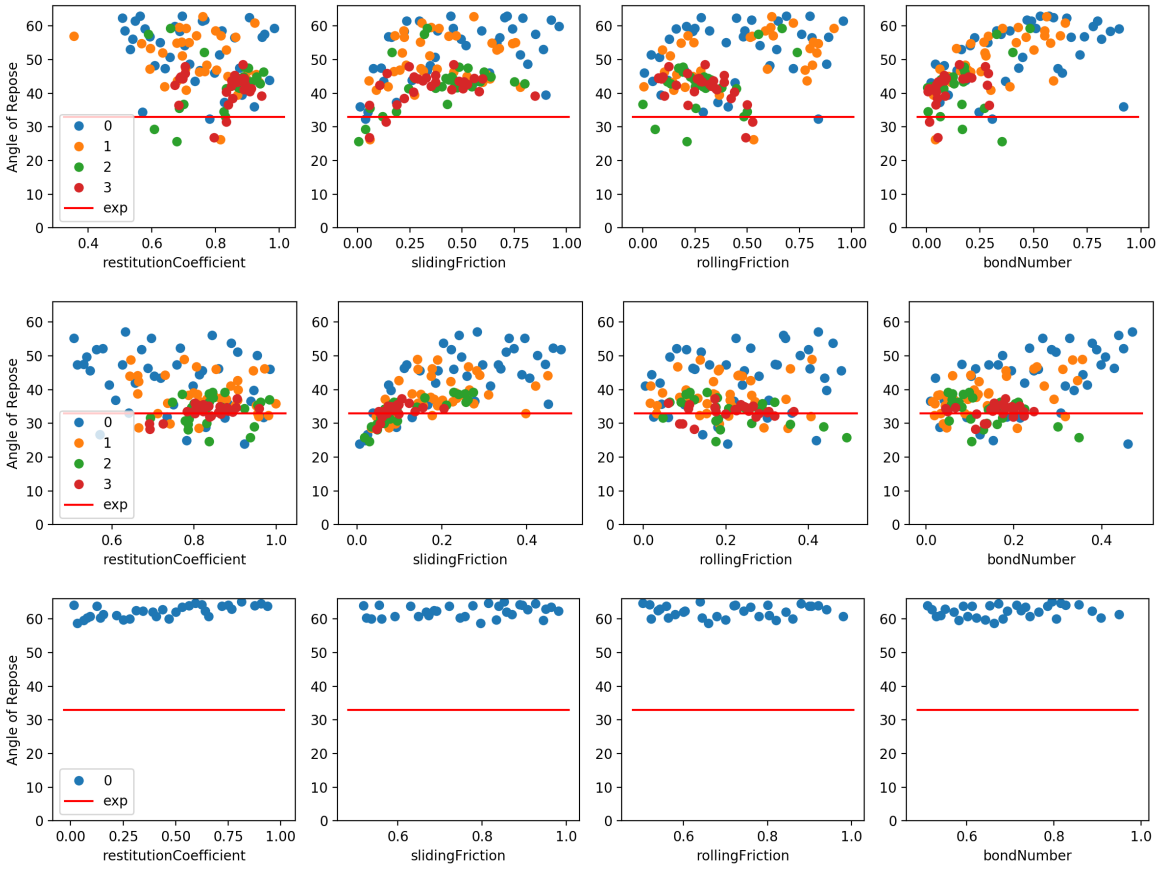

Figure 4: *From top to bottom,* attempt 1, 2, and 3 at calibrating Eskal with GL. Scatter dots with different colors denote each iteration, and red line denotes experimental value.

4.2 Quartz sand

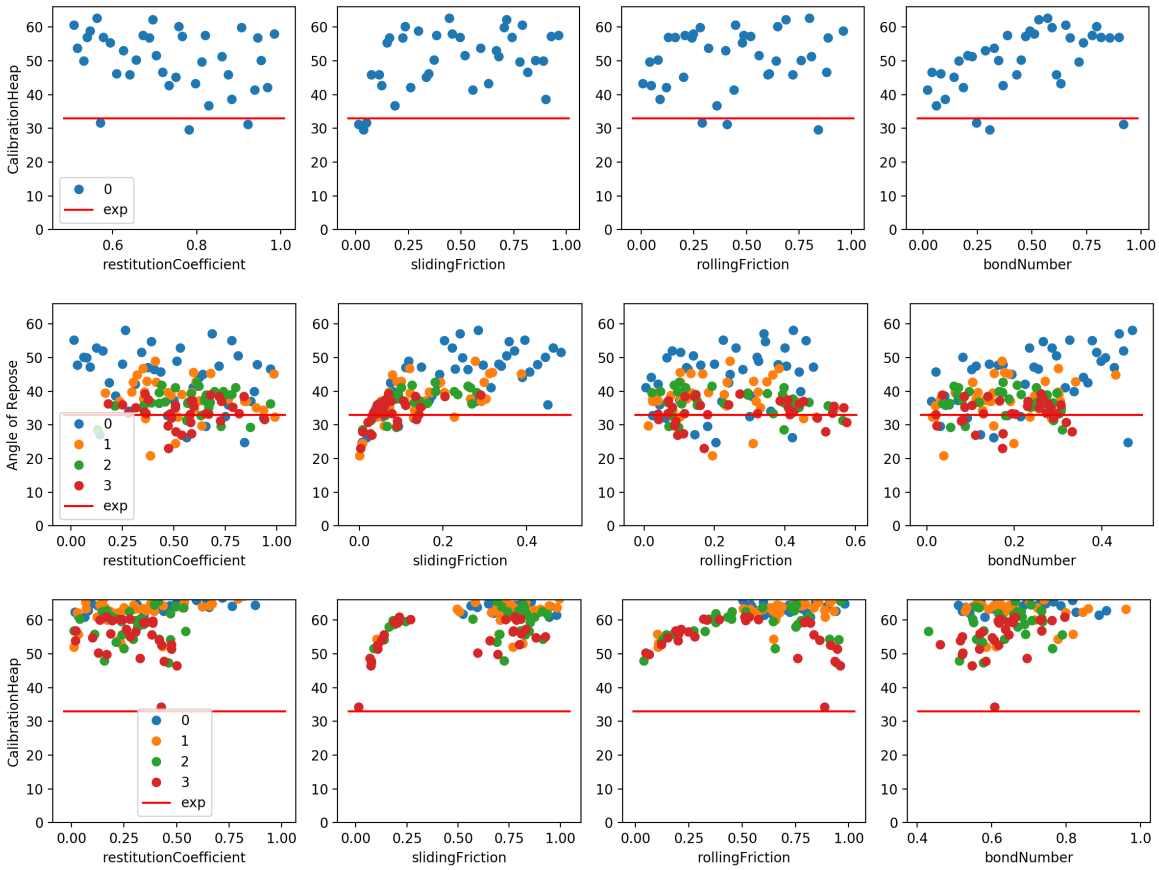

Figure 5: *From top to bottom,* attempt 1, 2, and 3 at calibrating quartz sand with GL. Scatter dots with different colors denote each iteration, and red line denotes experimental value.

Throughout multiple calibration routines with GrainLearning, it is identified that the initial guessing range played a key role in the ability of GL to idenfity the correct set of micro-parameters.

References

[1] A. Fernandes, H. Gomes, E. M. B. Campello, P. Pimenta, A fluid-particle interaction method for the simulation of particle-laden fluid problems, 2017. doi:10.20906/CPS/CILAMCE2017-0139.

[2] A. B. Yu, F. Bassani, G. L. Liedl, P. Wyder, Powder Processing: Models and Simulations, Encyclopedia of Condensed Matter Physics, Elsevier, Oxford, 2005, pp. 401–414. doi:10.1016/B0-12-369401-9/00556-8.

[3] E. S. Oran, J. P. Boris, R. A. Meyers, Encyclopedia of Physical Science and Technology (Third Edition), Academic Press, New York, 2002, Ch. Fluid Dynamics, pp. 31–43. doi:10.1016/B0-12-227410-5/00248-9.

[4] R. M. Nedderman, Statics and Kinematics of Granular Materials, Cambridge University Press, Cambridge, 1992, Ch. Introduction, pp. 1–6. doi:10.1017/CBO9780511600043.002.

[5] X. Weng, Modeling of complex hydraulic fractures in naturally fractured formation, Journal of Unconventional Oil and Gas Resources 9 (2015) 114–135. doi:10.1016/j.juogr.2014.07.001.

[6] P. A. Cundall, O. D. L. Strack, A discrete numerical model for granular assemblies, Géotechnique 29 (1) (1979) 47–65. doi:10.1680/geot.1979.29.1.47.

[7] L. Benvenuti, C. Kloss, S. Pirker, Identification of dem simulation parameters by artificial neural networks and bulk experiments, Powder Technology 291 (2016) 456–465. doi:10.1016/j.powtec.2016.01.003.

[8] P. He, Y. Fan, B. Pan, Y. Zhu, J. Liu, D. Zhu, Calibration and verification of dynamic particle flow parameters by the back-propagation neural network based on the genetic algorithm: Recycled polyurethane powder, Materials 12 (20) (2019). doi:10.3390/ma12203350.

[9] D. Schiochet Nasato, R. Queiroz Albuquerque, H. Briesen, Predicting the behavior of granules of complex shapes using coarse-grained particles and artificial neural networks, Powder Technology 383 (2021) 328–335. doi:10.1016/j.powtec.2021.01.029.

[10] F. Ye, C. Wheeler, B. Chen, J. Hu, K. Chen, W. Chen, Calibration and verification of dem parameters for dynamic particle flow conditions using a backpropagation neural network, Advanced Powder Technology 30 (2) (2019) 292–301. doi:10.1016/j.apt.2018.11.005.

[11] H. Q. Do, A. M. Aragón, D. L. Schott, A calibration framework for discrete element model parameters using genetic algorithms, Advanced Powder Technology 29 (6) (2018) 1393–1403. doi:10.1016/j.apt.2018.03.001.

[12] H. Cheng, T. Shuku, K. Thoeni, H. Yamamoto, Probabilistic calibration of discrete element simulations using the sequential quasi-monte carlo filter, Granular Matter 20 (1) (2018) 11. doi:10.1007/s10035-017-0781-y.

[13] H. M. Beakawi Al-Hashemi, O. S. Baghabra Al-Amoudi, A review on the angle of repose of granular materials, Powder Technology 330 (2018) 397–417. doi:10.1016/j.powtec.2018.02.003.

[14] M. Rackl, F. E. Grötsch, 3d scans, angles of repose and bulk densities of 108 bulk material heaps, Scientific Data 5 (1) (2018) 180102. doi:10.1038/sdata.2018.102.

[15] T. F. Teferra, Chapter 3 - engineering properties of food materials, in: M. Kutz (Ed.), Handbook of Farm, Dairy and Food Machinery Engineering (Third Edition), third edition Edition, Academic Press, 2019, pp. 45–89. doi:doi.org/10.1016/B978-0-12-814803-7.00003-8.

[16] T. Weinhart, L. Orefice, M. Post, M. P. van Schrojenstein Lantman, I. F. Denissen, D. R. Tunuguntla, J. Tsang, H. Cheng, M. Y. Shaheen, H. Shi, P. Rapino, E. Grannonio, N. Losacco, J. Barbosa, L. Jing, J. E. Alvarez Naranjo, S. Roy, W. K. den Otter, A. R. Thornton, Fast, flexible particle simulations — an introduction to mercurydpm, Computer Physics Communications 249 (2020) 107129. doi:10.1016/j.cpc.2019.107129.

[17] S. Luding, Cohesive, frictional powders: contact models for tension, Granular Matter 10 (4) (2008) 235. doi:10.1007/s10035-008-0099-x.

[18] H. A. Navarro, M. P. de Souza Braun, Determination of the normal spring stiffness coefficient in the linear spring–dashpot contact model of discrete element method, Powder Technology 246 (2013) 707–722. doi:10.1016/j.powtec.2013.05.049.

[19] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, TensorFlow: Large-scale machine learning on heterogeneous systems (2015). doi:10.5281/zenodo.4724125. URL https://www.tensorflow.org/

[20] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016, http://www.deeplearningbook.org.